1969

An optimal sequence sensitivity analysis for the flow shop scheduling model

Milan M. Sowis
Lehigh University

AN OPTIMAL SEQUENCE SENSITIVITY ANALYSIS FOR THE FLOW SHOP SCHEDULING MODEL

by

M. M. Sowis

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

Lehigh University
1969

CERTIFICATE OF APPROVAL

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

May 13, 1969
Date

Sutton Munro
Professor in Charge

F. E. Kane
Chairman of the Department
of Industrial Engineering

ACKNOWLEDGEMENTS

The author wishes to thank Professor Sutton Monro of Lehigh University for his timely advice and guidance, and George Schultz of the Western Electric Electric Company, Engineering Research Center, for his technical guidance in this effort.

TABLE OF CONTENTS

ABSTRACT

The problem of constructing an optimal schedule or sequence, given that there are n jobs and m machines, has been intensively studied by many investigators. Analysis of problems posed by perturbations which occur after computation of the optimal sequence have not been reported in the literature. This paper presents an analysis of the effects on the optimum sequence of changing the time of a single operation after an optimal schedule has been determined but before the schedule has actually commenced. The scheduling model treated is the static flow shop where the objective function is to minimize overall processing time.

I. INTRODUCTION

The problem of constructing an optimal schedule or sequence, given that there are n jobs and m machines, has been intensively studied by many investigators. The extant literature is replete with mathematical algorithms which can be used to obtain optimal or near optimal solutions to the static case of the scheduling problem. In the static case of the scheduling problem, all of the n jobs are assumed to be available prior to the start of the scheduling function. Each of the n jobs consists of m operations, or one operation on each of the m machines. In the static case, the operation times are assumed to be deterministic and are known a priori. In the dynamic scheduling problem, new jobs are allowed to enter the system after the start of the scheduling function. Operation times are assumed to be probabilistic and are known a posteriori. Thus, solutions to dynamic problems are based upon probability and queueing theory while solutions to static problems are more algebraic in nature.

Very little investigation has been attempted in the area of transition between the static and dynamic cases. Suppose that we are confronted with a static scheduling problem and that we have used a mathematical set of rules to construct an optimal schedule for some criterion which we want to optimize. One of three possible situations can now occur:

1. Everything goes smoothly and exactly as anticipated. This poses no problem and is also highly unlikely.
2. While the schedule is in progress, there occurs some sort of

breakdown situation, i.e., a perturbation enters the system.

3. Prior to the actual start of the original schedule, but <u>after</u> its computation, some new information becomes available.

The purpose of this thesis is twofold. Firstly, its purpose is to identify and categorize the possible problems that may arise. Secondly, one of these particular problems is selected for detailed analysis. This will be discussed in a later portion of this paper.

II. DESCRIPTION OF THE SCHEDULING PROBLEM

We want to schedule n jobs through m machines in such a manner as to optimize an objective function or scheduling criterion. The solution to the problem is a sequential ordering of the n jobs for each of the m machines and, as such, is simply a Gantt chart which optimizes the objective function.

All scheduling problems can be divided into two basic groups.

Group I - The Static Case.

There are n jobs, each of which requires processing on some or all of m machines. All of the n jobs are assumed to be available at "time zero," that is, no future arrivals are permitted.

Group II - The Dynamic Case

New jobs are allowed to enter the shop. Each time anyone of the m machines completes an operation, a decision must be made as to which of the available jobs should be assigned to the machine.

The static case is the subject of this thesis. When solving the static scheduling problem, two additional assumptions are usually made. These are:

1. only one job can be processed on a machine at a time;
2. all of the processing times are known a priori and are deterministic.

The partial relaxation of this second assumption poses the problem of this thesis.

Within the static case there are basically three categories. These are:

1. Flow Shop

In a flow shop the order of processing on the m machines is the same for each of the n jobs. For example, if we have two jobs, A and B, and two machines, 1 and 2, and the order of processing for each job is machine 1, then machine 2, each operation of job A must be completed on machine 1 before it can start on machine 2. There are n! possible sequences. In the above example there are two; Either A is processed first on each machine or B is.

This case is sometimes referred to in the literature as "no passing allowed." It is not required that every job have an operation on every machine nor must all work enter on a single machine, or leave from a single machine. This means that some jobs may have zero time on some machines but the sequence, 1, 2, . . ., m must be preserved.

2. Job Shop (Randomly Routed)

In this case, there is no specified machine sequence required for any job as it passes through the shop. For example, if there are two jobs, A and B, and two machines, 1, 2, we can start job A on machine 1 and job B on machine 2 simultaneously. In this case the number of possible sequences is $(n!)^m$. For the example there are four. These are:

MACH 1	A B	1 B A	1 A B	1 B A
MACH 2	A B	2 B A	2 B A	2 A B

This case is sometimes referred to as "passing allowed." When there are a large number of jobs and few machines, the problem becomes trivial, in that queues of waiting jobs will form at each machine. One need only schedule the jobs one right after the other according to

some selection rule.

3. Job Shop (Arbitrary Routing) (General n/m Problem)

In this case, each one of the n jobs has a particular machine sequence assigned to it. There may be a group of jobs with the same sequence and there may be some jobs which can be randomly routed as in case 2. Also there may be "directly precede" and "precede" relationships. An example of a "directly precede" relationship is one where job 1 must be processed on machine 1 directly before it goes on machine 2. An example of a "precede" relationship is one where job 1 must be processed on machine 1 before it goes to any other machine. It does not matter to which machine it goes next, but it cannot go to any machine before machine 1 processing is completed. This case is sometimes referred to as the general n/m job shop problem. It allows combinations of jobs with predetermined machine sequences and also random sequences. The flow shop problem to be treated in this thesis consists entirely of "directly precede" relationships.

There are any number of constraints which may or may not be applicable to a particular problem. Some of these are:

1. General precedence constraints (directly precede and precede relationships).
2. Setup time is sequence dependent or independent. If the setup times are independent of sequence, the setup time can be absorbed into the processing time.
3. Parallel machines. This means that more than one machine of the same type is available.

4. Splitting of jobs allowed or not allowed. This means that an individual job may be split over more than one machine (if there are parallel machines) or that a job may be started on a machine, interrupted, and then resumed at a later point in time.

5. Constraints imposed by specifying due dates.

6. Constraints imposed by having priorities for different classes of jobs.

There are a multitude of criteria by which schedules can be evaluated. Some of these are:

1. Minimize either the average amount of time or the maximum amount of time by which due dates are missed.
2. Minimize setup times (costs).
3. Maximize machine utilization where machine utilization is the ratio of available machine time that is used to total available time.
4. Meet certain priorities when there are preferred classes of jobs.
5. Minimize production costs.
6. Spread work uniformly over available machines.
7. Minimize average completion time.
8. Optimize some inventory policy.
9. Minimize flow time or "makespan" (schedule length).

The last of these is easily the most often used and much of the literature has been concerned with minimizing the total time for all

of the jobs to process through all of the machines.

III. SOLUTIONS TO THE STATIC SCHEDULING PROBLEM

If there are n jobs and only one machine and set-up time is sequence independent, the problem of minimizing makespan is trivial. Obviously makespan is independent of sequence. However, the average flow time is minimized by sequencing the jobs in order of increasing processing time, i.e., using the shortest processing time (SPT) rule. It can also be shown that longest processing time (LPT) sequencing maximizes whatever SPT minimizes. Such pairs of scheduling procedures which produce opposite sequences are called antithetic. The case of n jobs and one machine can be analytically solved for almost every objective function. Unfortunately, this is not true when the number of machines is greater than one. When the case of two or more machines is considered, the static case can be further classified into two different categories:

General Flow Shop Case

Hereafter, when speaking of a "solution" to a problem, this means minimizing makespan unless otherwise stated. In the general flow shop problem of n jobs and m machines, there are solutions available for only three cases. These are:

a. n jobs, 2 machines A and B, all jobs processed in order AB. The solution was developed by Johnson[13].

b. n jobs, 3 machines A, B, and C, and all jobs processed in order A B C. In addition, two other restrictions must be met; (1) smallest processing time on A is greater than or equal to the largest processing time on B and (2) smallest

processing time on C is greater than or equal to largest processing time on B. This solution was also developed by Johnson[13].

c. 2 jobs m machines. Each job must process through the m machines in a given order (order used need not be the same for both jobs). The solution was developed by Akers[1].

These are the only cases to which an analytic solution is known. There has been no analytical method developed for minimizing mean flow time, even for the two machine case.

There are, however, methods which are called non-analytic or computational that can be used to solve the flow shop problem. One of these is the Branch and Bound method which is discussed in more detail in another section of this thesis. Conway, Maxwell, and Miller[6] give an excellent discussion of the state of the art, not only relating to the flow shop problem but to the entire scheduling field. Wagner[23], with Story[22] and Giglio[9] has used integer programming to solve the problem, as has Manne[20]. Dudek and Teuton[7] proposed an effective combinatorial approach which is not, however, a generally theoretical one since Karush[14] subsequently posed a counterexample. Heller[10] proposed an effective sampling procedure based on statistical methods for use on large problems. In general, even the computational methods are impractical for very large problems.

General n/m Job Shop Case

The only case for which a mathematical solution is known is the two machine problem with the special restriction that each job can have at most two operations. This has been proposed by Jackson[12].

According to Conway, Maxwell, and Miller, "many proficient people have considered this problem, and all have come away essentially empty handed."

Some discussion is warranted on what is meant by a solution to a scheduling problem. There is a solution to every scheduling problem, since complete enumeration of all possible sequence permutations through all machines will yield on optimum sequence. In many cases this is impractical even with the use of high speed digital computers. What is meant when we say there is no solution is that there is no "quick", analytical, mathema tical method of producing one. There are, however, certain "computational" techniques available which do yield solutions to large classes of problems.

Thus far we have subdivided the scheduling problem, as far as solutions are concerned, into the static and dynamic cases and further subdivided the static case into the flow shop and the general job shop cases. The remaining portion of this thesis is concerned with the flow shop case.

A solution to a flow shop problem is defined as a permutation of the n jobs called a sequence. This sequence must be maintained on each of the m machines. Any feasible sequence is a solution in that it does not violate the flow shop constraint of no passing. A solution which minimizes the overall makespan is called an optimal solution.

Scheduling Approaches

There are basically two approaches to scheduling:

1. Schedule according to an analytic algorithm which will op-

timize the required criterion.

2. Schedule according to a heuristic dispatching rule. A dispatching rule is defined as one which selects a job (or operation) for a particular machine only when that machine becomes available or idle.

The second approach does not readily lend itself to the no passing flow shop model under consideration. The reason is that once an operation is selected for machine 1, all of the rest of the machines are automatically scheduled also. The net effect is to exercise the selection process for machine 1 only. A second reason that this approach is less appropriate for the flow shop is the absence of the due date constraints which most dispatching disciplines utilize in the selection process.

The heuristic approach, when applied to scheduling problems, can be called a "look ahead" approach, that is, from the list or queue of jobs available for processing, one is selected to be processed next on a machine which becomes idle. The selection is made based upon a heuristic rule such as selecting a job based on

1. Total slack,
2. Shortest imminent operation,
3. Slack per operation,
4. Various other versions and modifications of these rules which are possible.

Gere[8] investigates the effectiveness of various priority dispatching rules. As with most heuristic approaches, his work is applicable primarily to the dynamic case and was designed primarily for the due date problem vis-a-vis the optimal makespan problem. Conway[3,4] and with Maxwell[5], has done considerable work in this same area. The heuristic approach is presented here for background purposes only and, as previously explained, is not usually appropriate for consideration in flow shop scheduling.

IV. DISCUSSION OF PERTURBATIONS IN THE FLOW SHOW MODEL

In this section we will merely list some of the possible "schedule degradations" which may occur in our deterministic, static, a priori, flow shop model. As such they represent the most elementary transitions from the static to the dynamic case. This section will merely serve to categorize some exogenous developments which might occur and some of the inferences which might arise as a result. Each of them might be the subject of a thorough analytic study. Some discussion is included about the machine breakdown situation.

Machine Breakdown

A machine will be out of service for a known interval of time. Even for the flow shop case, there are many possible situations. For instance,

a. the machine on which the breakdown occurs is currently idle.

b. the machine on which the breakdown occurs is currently processing an operation.

For each of these possibilities

c. the downtime may last until the current schedule runs out or

d. the machine may become available again before the current schedule runs out.

In the case where an operation of job i is being processed on machine j at the time of breakdown, one of the following may be required:

e. job i must be started over on machine l (no resume, start over).

f. the operation of job i which was being processed at the time

of the breakdown must start over on machine j,

g. the interrupted operation may resume where it was interrupted when machine j becomes available again.

Further, the aborted operation can follow one of the restart requirements (e, f, g) either

h. immediately or

i. not until a certain period of time has elapsed. If there is elapsed time until machine j becomes available, again there are two possibilities:

j. other jobs can be processed, i.e., the machine is out of serivces to one job only or

k. the machine is out of service to all jobs.

The length of the elapsed time may extend until

l. the remainder of the schedule "runs out" or

m. some point in time before the schedule "runs out."

As can be seen, there are many combinations which can occur. Each of these combinations creates a new optimization problem for the uncompleted portion of the schedule for which the existing sequence may or may not be optimal.

It is apparent that, if an operation currently in process at the time of the interruption is on the schedule critical path, (to be discussed subsequently) the overall makespan must be lengthened by the duration of the breakdown if the original sequence is maintained. However, it may be true that the portion of the problem which must be run again has an optimal sequence different from that of the disrupted schedule. Hence, the overall makespan will be lengthened by

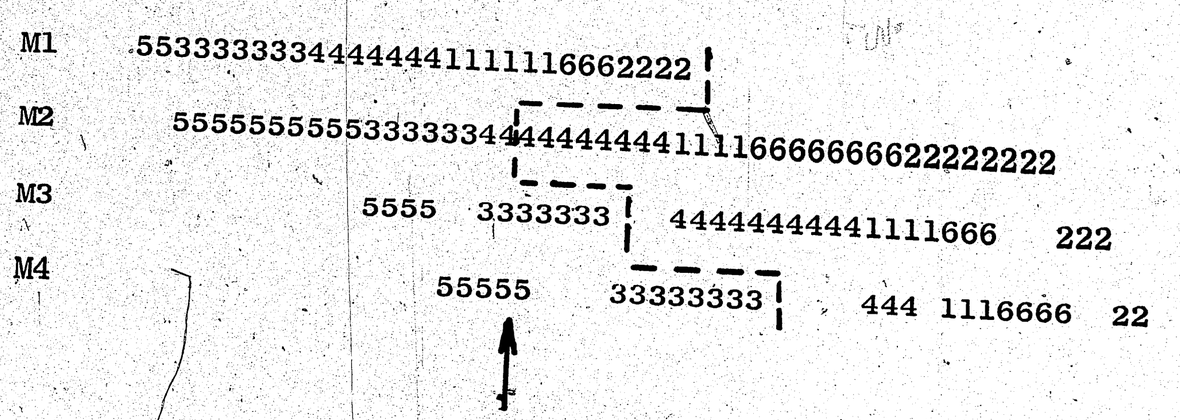
an amount shorter than the duration of the breakdown.

In general, if there are m machines, and, at some point in time, the i^{th} machine breaks down while processing the j^{th} operation on that machine (not necessarily the job numbered j), the original schedule can proceed uninterrupted on machines 1,...,i-1. It must halt on machine i. On machines i+1,...,m, the original schedule can proceed to the j^{th} operation. In other words, those jobs not yet completed on machine i cannot go on to machine i+1,...,m until the breakdown situation is corrected.

As an example, suppose job 4 on machine 2 encountered a breakdown condition at time 20 (See Fig. 1 on following page). Two time units of processing have been completed (condition b). Further suppose the machine did not become available again until time 53, when the remainder of the schedule has been completed (condition c). In other words, all operations on machine 1 can run to completion and jobs 5 and 3 on machines 3 and 4 can also be completed. If at that point condition (e) holds, then the new problem to be scheduled is

	M1	M2	M3	M4
J1	0	4	4	3
J2	0	8	3	2
J4	7	10	10	3
J6	0	8	3	4

However, if condition (f) holds, then we have



Breakdown (time = 20)

(Operations to left of dotted line can be completed)

Figure 1

	M1	M2	M3	M4
J1	0	4	4	3
J2	0	8	3	2
J4	0	10	10	3
J6	0	8	3	4

If condition (g) is applicable, then

	M1	M2	M3	M4
J1	0	4	4	3
J2	0	8	3	2
J3	0	8	10	3
J4	0	8	3	4

Any of the new problems may yield an optimum sequence which differs from the original one which is (4, 1, 6, 2).

Change in Job Priorities

We have been concerned so far with optimizing overall makespan and not with any due dates. If such constraints are added to the model after the optimal schedule has been completed, a rescheduling problem is formulated which has many possibilities. A job within the schedule may be assigned a completion time from without the system, as by management. This is termed an exogenous due date. The schedule must be revamped to meet this date (or perhaps to miss it by the minimum amount possible).

Further, it may be required for some reason (such as commitments to customers after establishment of the original schedule)

to maintain the completion dates which have been tacitly assigned by the generation of the optimal makespan schedule itself endogenous due dates). This would further complicate any rescheduling process.

Lack of Component Parts

In an assembly type of operation, certain piece parts are required to complete an operation of a job before the job can proceed to the next station. A shortage of these parts might occur. The lack of piece parts is analogous to a machine breakdown in that the station (machine) will remain idle for a period of time. It differs in that another operation might be selected in place of the one on which the shortage occurred.

New Job(s) Enters the System

Two of the alternatives facing the rescheduling function are:

a. Add new job to tail end of current schedule

b. Insert the new job into the existing sequence

In addition, the new job(s) may enter the system with an exogenous due date.

Operation Takes Longer Than Original Estimate (Original Estimate was Considered Deterministic)

There are two possible actions:

a. No action, i.e., let the operation continue to run "overtime" until completion on that machine.

b. Stop operation on that machine at the point where the excess time is equal to the slack allowed for that operation by the original schedule.

The analysis might be the same as in case I (machine breakdown) if the operation is allowed to finish the current operation. During the interval that the operation exceeds its allotted time, the machine on which it is running is, in effect, out of service to all other jobs.

Action (b) would preserve integrity of original schedule, assuming that the job in trouble could "go on" to subsequent machines as originally scheduled (except job in question would have to return to that machine for completion later).

In a flow shop, action (b) would remove the job in question from the schedule of all subsequent machines creating "gaps" in the schedule. The remaining jobs could be rescheduled according to the original rule.

If a job takes less than the "deterministic estimate," and that job is on the schedule critical path, the overall schedule will be reduced by that amount. If the job is not on the critical path, then the effect will be an increase in slack for that operation.

Cancellation of a Job(s)

Machine Running Slow

A machine is running slow and turning out all jobs in a

longer time but at a uniform rate.

Job Running Slow

A job is running slow on all machines, i.e., all operations for a particular job are "running longer" and at a uniform rate.

As previously stated, the preceding paragraphs describe situations to which this thesis is not analytically addressed. However, these situations do indicate the complexity and variability of possibilities which can occur within the framework of the simple flow shop model. They are presented primarily to provide the reader with an insight into the ramifications which are possible.

V. ACTUAL PROBLEM TO BE INVESTIGATED

The specific situation which this thesis investigates is the following:

We are given a schedule that minimizes the makespan for a flow shop which has been assumed to be a deterministic flow shop. This schedule is some permutation of the numbers of the job set, called an optimal sequence.

After the schedule is constructed, but prior to the start of that schedule, we are told that one of the operation times has taken on a different value. A decision must be made as to whether one should reschedule or not. Of course, one could always re-enter the original algorithm again. The question to be decided, however, is is "Can one make some sort of decision without going through the algorithm again?" In other words, the determination of a new optimal sequence requires (1) time to prepare and enter the new information and (2) time to run the algorithm on the computer.

It is true that we may change an operation time and immediately decide whether our original makespan is affected or not by examining the slack for that particular operation. One question to be answered is, "By how much can an operation time be increased before another sequence becomes optimal?" Another question is "How much an operation time can be decreased before another sequence becomes optimal?"

Perhaps it can be determined with what probability an optimal

sequence already known will remain an optimal sequence, given a certain range in a particular operation time. Perhaps certain heuristic decision rules can be developed. Thus, the questions to which the scheduler requires an answer when confronted with this situation are:

1. Is the original optimal makespan still optimal?
2. Is the original sequence still optimal?

The goal of this thesis is to provide some insight to the problem so that question (2) can be answered with some degree of assurance without reentering data and rerunning the algorithm used to generate the original schedule.

VI. BRANCH AND BOUND ALGORITHM

The problem under investigation in this thesis can be thought of as a sensitivity analysis. By varying the input parameters, which are the operation times of the jobs on the machines, one can observe the effect on the optimal sequence. The particular algorithm selected to construct the optimal schedules used in this study is known as a Branch and Bound algorithm.

Branch and Bound is a structured search of a solution space for an optimum solution. The branching provides the enumeration of the problem and the bounding limits the enumeration to that required to obtain an optimal solution. Branch and Bound is a self-generating technique which, once started, branches according to the bounds established by examination of previous branches. The efficiency of the technique is its ability to limit the enumeration of the problem to that required to obtain the desired solution. Without bounding the process would continue until $n!$ branches were created.

A branch and bound algorithm ~~is~~ defined by specifying

1. a branching rule
2. a bounding problem to determine lower bounds for new modes
3. a rule for determining which mode to branch from at any state in the algorithm
4. a termination condition (recognizing when a final node

contains an optimum solution)

An excellent description of how the Branch and Bound technique works is found in a paper by Agin[1]. Lawler and Wood[15] present a survey of applications of the technique. The procedure was originally developed by Little et al.[16] for use on the well-known traveling salesman problem.

The sensitivity analysis under discussion here is really independent of the particular algorithm used to solve the problem. We are only concerned with the effect of varying one of the operation times upon the optimal sequence and are not concerned with what steps the algorithm actually used to determine the optimal sequence. Consequently, the Branch and Bound algorithm need not have been the algorithm used to develop the solution.

However, the Branch and Bound algorithm lends itself extremely well to the flow shop sequencing problem, since it is a combinatorial problem with a finite solution space. Lomnicki[17] has developed a solution for the three machine problem and with Brown[18] extended the procedure to the general m machine case. It is the Lomnicki and Brown algorithm which has been used in this study. Ignall and Schrage[11] and McMahon and Burton[21] have also developed similar approaches to the flow shop sequencing problem. A good detailed explanation of the algorithm with numerical examples is given by Long[19].

VII. DISCUSSION OF GANTT-PERT-SLACK

A most valuable tool for analyzing a schedule and the effects of changing operation times on a schedule is the slack matrix for that schedule. There is a direct relationship between an optimal sequence as represented by a Gantt chart and the well-known PERT chart. Consider the Gantt chart of Figure 2a, which represents a solution for a particular operation time matrix. Since the various operations are interdependent in that the different operations cannot start before other operations are completed, the Gantt chart can be represented as a network as shown in Figure 2b or in its more familiar format shown in Figure 2c. In Figure 1c a node (i,j) represents the start of job i on machine j. In general, where there are n jobs and m machines, the network will appear as in Figure 2d. Hence any flow shop solution can be represented as a PERT network and each operation has an associated slack associated with it. The slack for each operation can be calculated from the PERT diagram, proceeding from the terminal node backwards to the origin.

It is, however, unnecessary to develop a PERT network in order to make slack calculations. The preceding was presented merely to illustrate the equivalence of the Gantt chart and the PERT network in the case of a flow shop. The slack for any operation can be calculated from the Gantt chart itself. We start by assigning zero slack to the last operation on the last machine. From then

on, the slack of the i^{th} operation on machine j is equal to the minimum of

1. the sum of the idle time remaining before the $i+1^{st}$ operation starts on the j^{th} machine plus the slack of the $i+1^{st}$ operation on the j^{th} machine

and

2. the waiting time of operation i before it can start on the $j+1^{st}$ machine plus the slack of the i^{th} operation on the $j+1^{st}$ machine.

Expressed mathematically

$$S_{i,j} = \text{MIN} \begin{cases} I_{i,j} + S_{i+1,j} \\ W_{i,j+1} + S_{i,j+1} \end{cases}$$

where

$S_{i,j}$ = slack of operation i on machine j

$I_{i,j}$ = idle time following operation i on machine j

$W_{i,j}$ = length of time between completion of operation i on machine j-1 and the start of operation i on machine j.

The mathematical formula is much more formidable than the calculations themselves. Any operation time matrix has associated with it at least one optimal solution and, correspondingly, at least one slack matrix. Every flow shop solution has associated with it a unique slack matrix. Referring again to Figure 2a, the operation time matrix is

JOB NO.	M1	M2	M3
1	2	4	5
2	3	1	1
3	8	3	7

and, for the particular solution shown, (1,3,2) the associated slack matrix is

JOB NO.	M1	M2	M3
1	0	2	2
2	6	6	0
3	0	0	0

Just as every PERT network has a critical path, so every flow shop solution (or Gantt chart) has exactly the same critical path. Every element in the slack matrix, for a particular solution, with a value of zero identifies an operation on the schedule critical path. Any increase in one of these critical operation times must, of necessity, extend the overall makespan by a corresponding amount. Similarly, any increase in any operation time which is in excess of its slack matrix value must increase the overall makespan by the same amount. The slack matrix plays a most vital role in our schedule analyses.

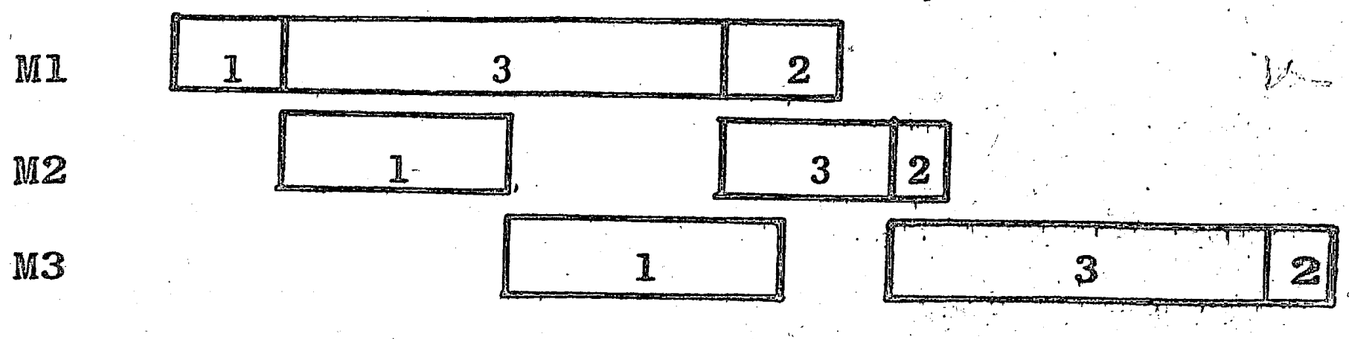


Figure 2a

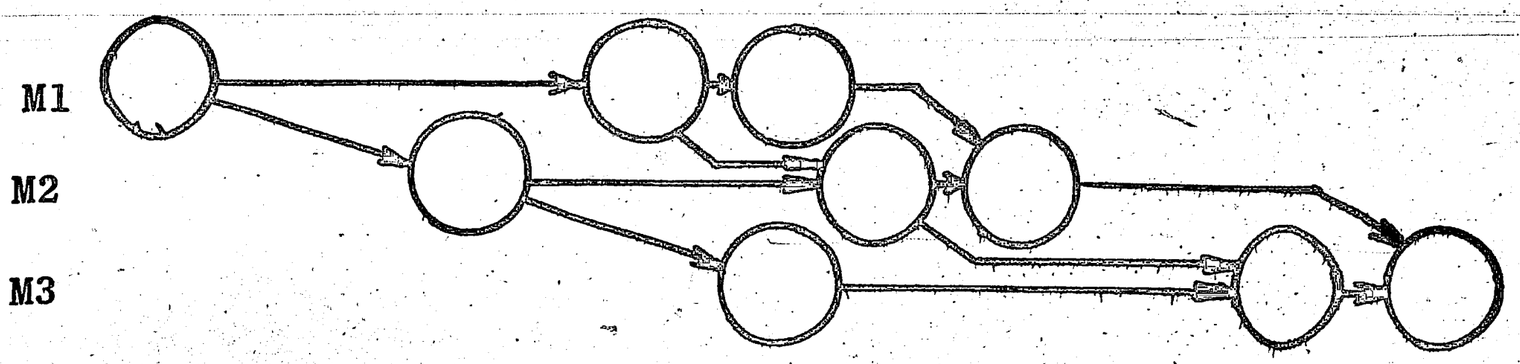


Figure 2b

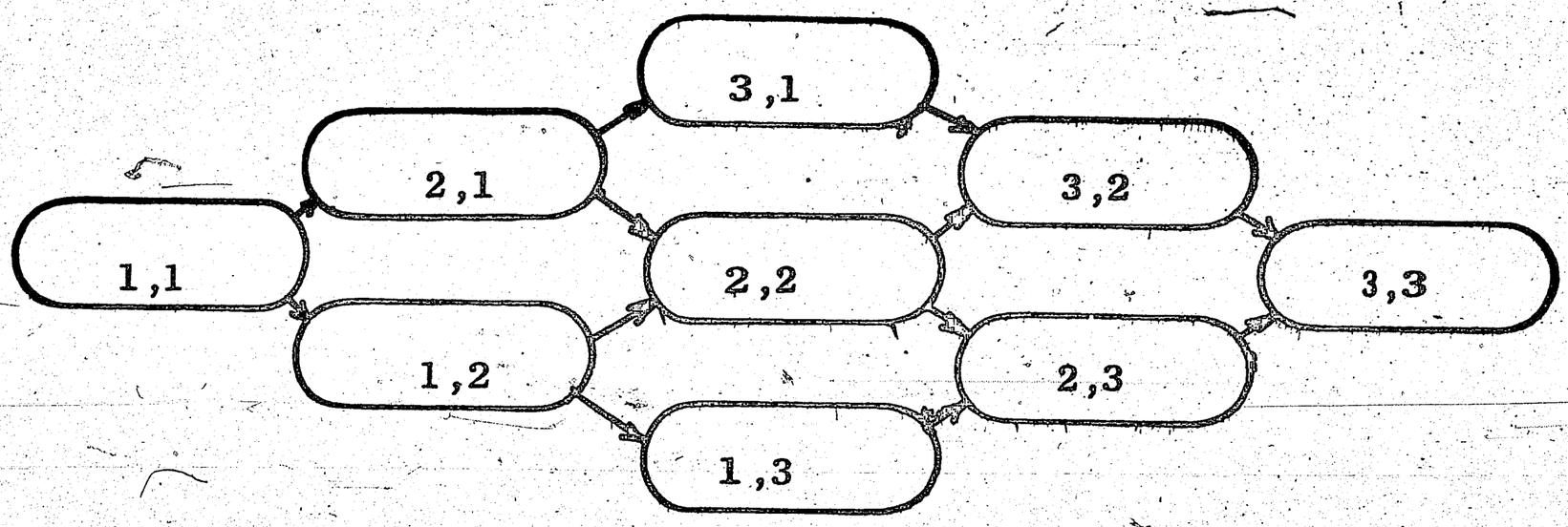
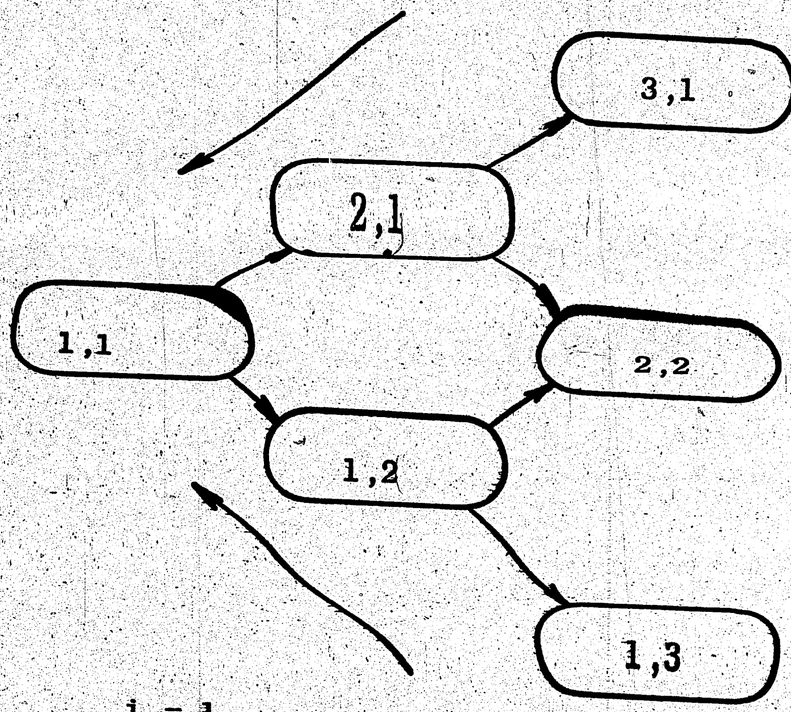


Figure 2c

i decreasing

from n to 1

j = 1



i = 1

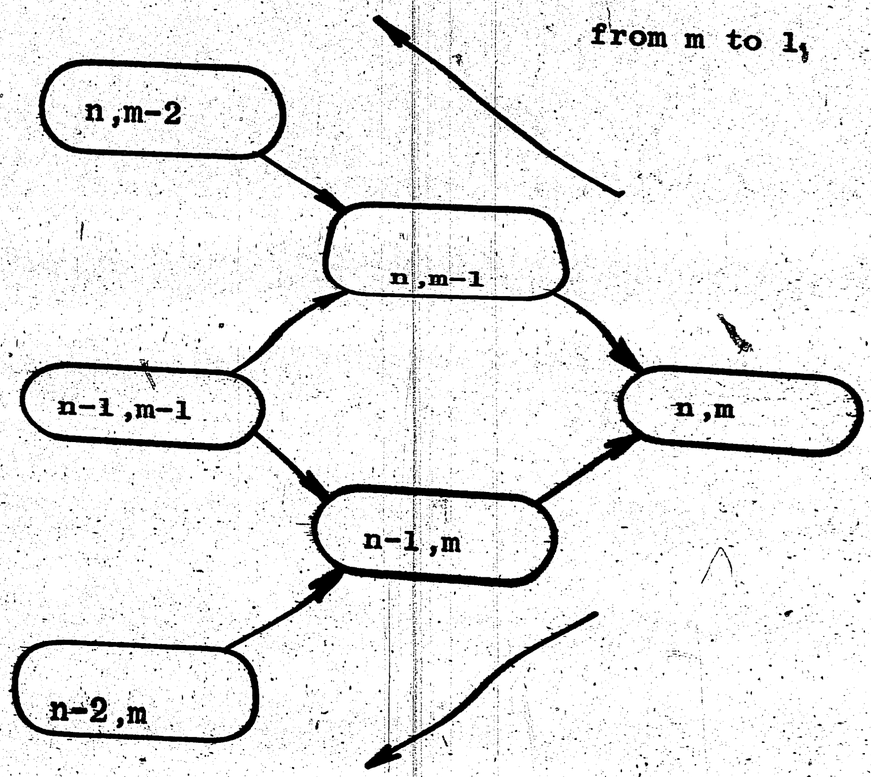
j decreasing

from m to 1

i = n

j decreasing

from m to 1



i decreasing

from n to 1

j = m

Figure 2d

VIII. DESIGN OF AN EXPERIMENTAL PROCEDURE

As explained previously, the primary objective of this thesis is to investigate the "permanence" of a known optimal sequence when one of the input matrix operation times is increased. As such, one can formulate the investigation as an experiment whose objective is to test a certain hypothesis. In this case the hypothesis to be tested is that some factors are significant as to whether a given optimal sequence remains optimal with increasing values of one of the input parameters.

The factors to be tested are

1. The relative magnitude of the operation time, e.g., short operations versus long operations.
2. The particular machine, e.g., the initial machine versus the last machine.
3. The amount of slack which the operation has. Included here is whether the operation lies on the schedule critical path.
4. The position of the job in the optimal sequence.
5. The "shape" of the operation time input matrix,
6. The range of the operation times.
7. The distribution from which the operation times are selected.

Before proceeding further, let us define the following terms:

M = input matrix of operation time

m_{ij} = the original value of an element of M

m'_{ij} = a changed value of m_{ij}

S = slack matrix

s_{ij} = an element of S

r = range of operation times in M

l = lower limit of the range of operation times

u = upper limit of the range of operation times

k = maximum number of increments

g = minimum makespan

A given input matrix, M, of operation times is specified. The dimensions of M are n x m where n is the number of rows and is equal to the number of jobs and m is the number of columns and equals the number of machines. Each element of M, m_{ij}, corresponds to the operation time of job i on machine j. The elements of M have a certain range of values with a lower limit, l, and an upper limit, u. A particular element of M is selected as the variable parameter while all others are held constant. The variable element is incremented by one time unit, thus creating a new input matrix, M', with m'_{ij} replacing m_{ij}. M' is used as input to the scheduling algorithm and an optimal sequence is produced. If this optimal sequence is the same as the previous one, m'_{ij} is again incremented by one and another optimal sequence is generated. The procedure is repeated until one of two events occur. Either

1. the optimal sequence does not change for a previously specified number of increments, k, or
2. the optimal sequence changes for a particular value of the variable parameter, m'_{ij}.

If condition (2) occurs, we can say that the optimal sequence "switches" when $m_{ij} = m'_{ij}$. If condition (1) occurs, we can say that the optimal sequence is "permanent" or that there is "no switch" in the optimal sequence. Another way of expressing this is to say that a particular operation is sensitive or not sensitive over a range of values. This procedure can be applied to all elements of the input matrix, one at a time. In so doing we can count the number of switches and non-switches and then determine how many operations the optimal sequence is sensitive to. By repeating this procedure for many input matrices, it should be possible to determine if there is any correlation between any of the factors to be tested for significance and the number of switches and/or non-switches. This in turn may provide some insight into the flow shop scheduling problem.

IX. USEFULNESS OF THE SLACK MATRIX

The incrementation of an operation time by one should not begin immediately at the value of $m_{ij} + 1$. As previously explained, every optimal sequence implies a unique Gantt chart which produces a unique slack matrix, S. Thus, a particular element m_{ij} can be incremented by a value of at least the value of its corresponding element in the slack matrix, s_{ij}, without changing either the optimal makespan or the optimal sequence. Consequently the incrementation should begin at the value $m_{ij} + s_{ij} + 1$ and continue until there is either a switch or the value $m_{ij} + s_{ij} + k$ has been reached and there has been no switch. Thus the analysis cannot even begin without the computation of the slack matrix.

To illustrate, consider the following problem. The input matrix, M, is

2	8	6	7
3	7	7	8
3	4	4	2
4	9	4	2

The optimal sequence is (3,2,1,4) with slack matrix

7	1	1	0
1	0	0	0
0	0	2	2
11	1	1	0

The optimal makespan is 38. This means that within this particular schedule, the incrementation of job 1 on machine 1 must begin at $m_{11} + s_{11} + 1$ or $2 + 7 + 1 = 10$. The incrementation of job 3 on machine 1 must begin at $m_{31} + s_{31} + 1$ or $3 + 0 + 1 = 4$, etc.

If m_{ij} is increased by $s_{ij} + 1$, and the optimal sequence is unique, then g must be increased by one and the original sequence must still be optimal. Consider the preceding example. If $m_{11} = 2$, as in the original problem, then $g = 38$. If $m_{11} = 9$, g is still 38. If $m_{11} = 10$, then g must be increased by one time unit to 39. But the original sequence of (3,2,1,4) will yield this value of g; hence it is still an optimal sequence. This argument of solution optimality cannot be extended for increments greater than 1, else all operations could be incremented by an infinite amount while retaining the same optimal sequence. What actually occurs is that certain elements m_{ij} will switch to different optimal sequences at some incremented value m'_{ij}.

X. ALTERNATE OPTIMUM SEQUENCES

The analysis as outlined in the previous paragraphs is valid only if the optimal sequence is unique, i.e., there exists only one sequence which yields a minimum makespan. The existence of more than one optimal sequence complicates the analysis. Consider the following example. The input matrix M is

4	5	8	8
7	1	9	4
7	9	8	5
6	8	2	1

with optimal sequence (2,1,3,4) , g of 41, and S =

0	2	2	2
0	2	2	6
0	0	0	0
6	3	3	0

Consider element m_{22}. If $m_{22} = 1 + 2 = 3$, the optimal makespan of 41 will still be retained. If $m_{22} = 4$, the optimal makespan obtained is still 41. The reason is that there exists another optimal sequence which yields g = 41. This sequence is (1,2,3,4) which has a slack matrix of

0	1	1	6
0	6	1	5
0	0	0	0
6	3	3	0

Here $s_{22} = 6$.

Therefore, the incrementation process must begin at a value equal to $m_{ij} + \max s_{ij} + 1$. In order to do this, one must be aware of all optimal sequences and respective slack matrices for the initial input matrix. The maximum slack for each element can then be computed and used as the starting point in the search for operation switches. This is the procedure used in this thesis.

The Branch and Bound algorithm as propounded by Lomnicki yields one optimal solution and, with a slight modification, will yield all optimal solutions for a given input matrix. The modification consists of setting the upper bound equal to the optimal makespan found when extracting the initial optimal solution, and then rerunning the algorithm. This will produce all optimal sequences. A computer program has been written to do this and to compute and store the slack matrices for each optimum sequence.

Extending the argument of solution optimality explained earlier, we can now conclude than an optimal solution is still optimal when $m'_{ij} = m_{ij} + \max s_{ij} + 1$ and that the g obtained using this value of m'_{ij} equals g + 1. If the increment is by more than one, another sequence may become optimal. Expressing this idea mathematically, if $m'_{ij} = m_{ij}$, the minimum makespan equals g. If $m'_{ij} = m_{ij} + \max s_{ij}$ the minimum makespan is still g. If $m'_{ij} = m_{ij} + \max s_{ij} + 1$, the minimum makespan becomes g + 1. If $m'_{ij} = m_{ij} + \max s_{ij} + 2$, and the minimum

makespan becomes g + 2, then <u>one</u> of the original optimal sequences must still be optimal. In general, if $m'_{ij} = m_{ij} + \max s_{ij} + k$ and the minimum makespan equals g + k, then <u>one</u> of the original optimal sequences is still optimal. If a point is reached where $m'_{ij} = m_{ij} + \max s_{ij} + k$ and the minimum makespan is g + k - 1, then the optimal sequence can be said to switch to a sequence which is not one of the original group.

This is the general method used in this thesis to discover the existence or nonexistence of the aforementioned sequence switches.

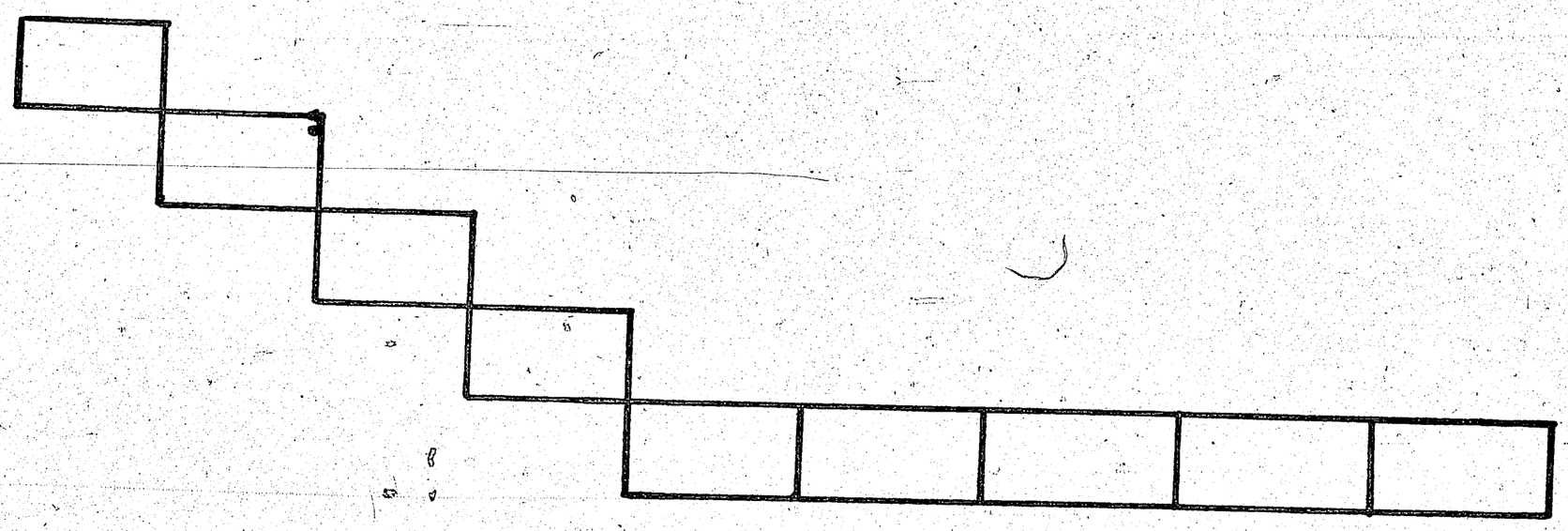
XI. SLACK MATRICES

By examining the slack matrices one can gain some insight into what the schedule "looks like" without actually looking at a Gantt chart. Some of the unique configurations which may arise are discussed.

If the first job in the optimal sequence has a slack of zero on the last machine, then the row corresponding to the first job and the last column (corresponding to the last machine) all have zero elements. In the schedule this means that the last machine has no slack or idle time and that the first job is "perpendicularly stacked" in the optimal schedule. Thus

X X X X 0
X X X X 0
0 0 0 0 0
X X X X 0
X X X X 0

corresponds to



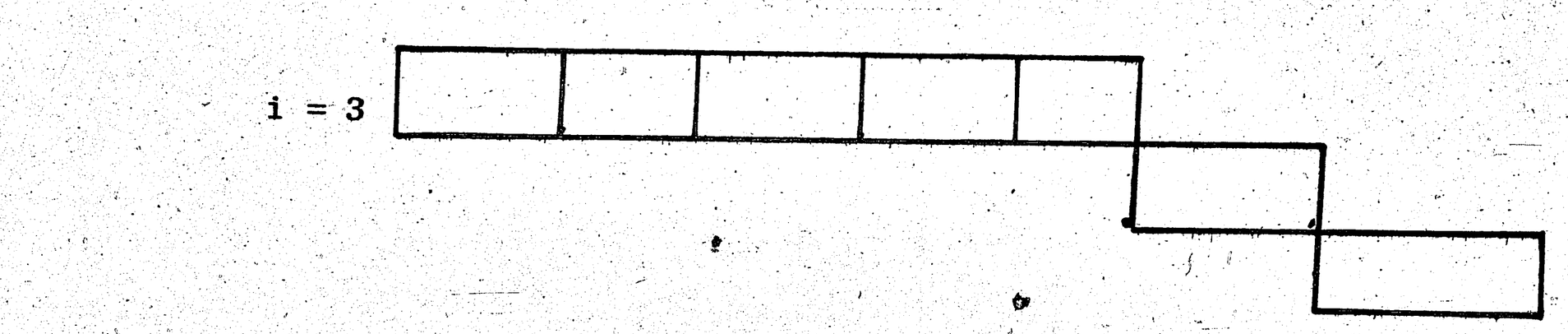
and vice versa.

If there is a column of zeros in column i of slack matrix, then there is no slack or idle time on the machine corresponding to that row. In addition, the last job in the sequence is "perpendicularly stacked" from machine i to the last machine.

Thus

X X 0 X X
X X 0 X X
X X 0 X X
X X 0 X X
X X 0 X X

corresponds to



SLACK TIME AND IDLE TIME

There is little correspondence between the amount of idle time and the amount of slack time in a schedule. For example consider the following 5 x 5 input matrix

7	4	9	4	5
9	1	6	6	5
3	3	6	10	6
2	7	6	5	9
7	6	8	3	3

This particular problem has five optimal sequences which yield a makespan equal to 50. Two of these are

(3,2,4,1,5) with slack matrix

2	2	0	2	0
0	1	2	0	0
0	0	0	0	0
0	0	0	0	0
2	2	0	0	0

and (3,4,1,2,5)

with slack matrix

2	4	2	4	0
2	8	2	2	0
0	0	0	0	0
2	2	2	1	0
2	2	2	2	0

If the Gantt charts were constructed it could be seen that the first schedule has twenty times units of idle time and the second has sixteen. However, the first schedule has a total of only 13 time units of slack for all of the jobs while the second has a total of 41 time units of slack. Thus, even though the first sequence has more machine idle time, the amount of slack time is considerably less. Yet both sequences yield the identical optimal makespan of 50.

It should be pointed out that the Branch and Bound method used to obtain this optimum sequence has as its goal the objective

of minimizing the makespan. According to Lomnicki , this is the same as minimizing the idle time. Yet, as we have just seen, two different optimal sequences have different amounts of idle time. The explanation to this apparent anomaly lies in the definition of idle time. The Branch and Bound algorithm, or for that matter, any algorithm which produces a minimum makespan sequence, minimizes the total idle time on all machines from time zero until time equal to the optimal makespan. This includes time during which a machine is idle before it starts on its first operation, and the time during which it is idle after it completes its last operation, as well as any time during which it is idle between operations. The definition of idle time, as previously given, considers only this inter-operation idle time.

XII. CRITERION FOR ALTERNATE OPTIMAL SEQUENCE EVALUATION

The difference in the amount of total slack between these two alternate optimal schedules leads immediately to another criterion for evaluating a schedule. As shown by the two slack matrices, the first sequence has a total of 13 slack units versus 41 for the second. Since both yield a makespan of 50, which should a scheduler select if given the opportunity? Obviously the second, since it provides a much greater protection against variation in the quasi deterministic operation times. Further, note that in the first sequence, the total slack is distributed over only seven of the 25 operations while the 41 units of slack in the second sequence are distributed over 16 different operations. The superiority of the second sequence is evident.

Another way of looking at the two sequences (and resulting schedules) is to realize that every zero in the slack matrix represents an operation on the critical path. In our examples, sequence one has 18 operations on the critical path (or paths) and sequence two, only 9. It is in general, desirable to minimize the number of critical paths and we are led to the same conclusion.

XIII. ANALYSIS OF RESULTS

A total of 550 matrices, M, comprising 11 groups of 50 each, were tested. The matrices were generated in sets of ten with a unique random number "seed," so that each group of 50 consists of five sets of ten each. The 11 groups tested are as follows:

Group	Size of Matrix (N x M)	Distribution	Parameters*
1	5 x 5	Uniform	1-9
2	4 x 4	Uniform	1-9
3	3 x 3	Uniform	1-9
4	4 x 6	Uniform	1-9
5	4 x 8	Uniform	1-9
6	6 x 3	Uniform	1-9
7	6 x 4	Uniform	1-9
8	5 x 5	Uniform	1-19
9	6 x 4	Uniform	1-19
10	5 x 5	Normal	N(10,2)
11	5 x 5	Normal	N(10,4)

The results are tabulated in Appendix A and summarized in Table I.

In all cases, K, the maximum number of increments, was equal to ten. The computer program, with provision to generate matrices from a uniform distribution, is listed as Appendix B.

---

*Values given are upper and lower limits of range for uniform distribution; mean and standard deviation for normal distribution.

TABLE I

Group	1	2	3	4	5	6	7	8	9	10	11
Size	5 x 5	4 x 4	3 x 3	4 x 6	4 x 8	6 x 3	6 x 4	5 x 5	4 x 6	5 x 5	5 x 5
Parameters	1-9	1-9	1-9	1-9	1-9	1-9	1-9	1-19	1-19	N(10,2)	N(10,4)
% SW	51.8	37.5	22.4	41.1	42.8	41.4	47.2	49.4	40.7	39.3	50.5
% NSW	48.2	62.5	77.6	58.9	57.2	58.6	52.8	50.6	59.3	60.7	49,5
% Short	40.9	47.6	66	43	39.8	48.3	46.8	38.7	40.6	36.3	43.3
% Medium	35	31.2	30	34.7	35.1	32.7	32.4	33.7	29.2	50.0	30.5
% Long	24.1	16.4	4	21.5	24.5	18.5	20.1	27.2	30.0	13.7	23
% Z	36	43.6	60.2	35.2	34.4	49.2	42.3	36.8	39.4	35.6	30.7
% NZ	64	56.4	39.8	64.8	65.6	50.8	56	63.2	60.6	64.4	69.3
% ZSW	63	45.7	29.5	48.7	51.7	39.7	57.7	53	48.6	48.6	70.3
% NZSW	45.5	32.3	11.7	24	38.2	43.1	40.7	47.1	35.6	34.2	41.7

See Appendix A for definition of abbreviations.

Effect of Relative Magnitude of Operation Time

Null Hypothesis: The relative magnitude of m_{ij} is not significant in determining whether an operation switches or not.

Consider the results obtained from Group 1. The total number of switches observed was 648. The range of operation times, 9, was divided into 3 classes, (a) short operations (1,2,3); (b) medium operations (4,5,6); (c) long operations (7,8,9); and the number of switches in each class was observed. If the null hypothesis is true, then we would expect 216 operations to switch in each class. The observed values from Appendix A are

123 SW* 265

456 SW* 227

789 SW* 156

Applying the Chi-square test with one degree of freedom yields a Chi-square statistic of

$$\frac{(265-216)^2}{216} + \frac{(227-216)^2}{216} + \frac{(156-216)^2}{216}$$

$$= \frac{49^2 + 11^2 + 60^2}{216} = \frac{2400 + 221 + 3600}{216}$$

$$= \frac{6221}{216} = 28.7$$

This is well outside the region of acceptance and we reject the null hypothesis.

*See Appendix A for definition of these terms.

The effect of the magnitude of operation time is most pronounced on smaller input matrices and tends to "damp out" as the size of the matrix is increased. Only 5% of the long operations in Group 3 (3 x 3) are classified as switches. This increases to 16.4% in Group 2 (4 x 4) and 24.1% in Group 1 (5 x 5). This might tend to indicate the effect diminishes when the input matrix is relatively large although no clear cut inference can be drawn.

Machine Effect

There is absolutely no evidence that a particular machine has any effect whatsoever on whether an operation switches or not. In every group tested, the likelihood of a switch appeared to be equal for the first machine, the last machine and all intermediate machines.

Effect of Slack

In every group tested, the percentage of "zero"* operations that switched was substantially greater than that of the "non-zero" operations (see Table 1). This result provides additional reason for more careful monitoring of critical path operations.

Effect of Position in Optimal Sequence

There appears to be no relationship between this factor and the number of switches. The analysis is complicated by the existence of alternate optimal solutions, i.e., a job can be first in one optimal sequence and not so in another. Consider the 50 matrices of Group 1. In 25 cases, there was either (1) only one optimal solution or,

*Zero operations are those with a maximum slack of zero, i.e., overall critical path operations.

(2) the same jobs were first (or last) in all optimal solutions. In the 25 initial jobs there were 77 switches and in the 25 terminal jobs there were 68. These results are typical and we must reject the significance of position in sequence.

Effect of "Shape" of M

Matrices in which there are more machines than jobs ($m > n$) appear to behave in much the same manner as matrices in which $n < m$ (see Table 1, Groups 4 and 5 versus Groups 6 and 7). There is a significant difference in one respect and that is the number of optimal solutions (NOP). Consider the following results.

Group	NOP	NOP/50
4 (4 x 6)	107	2.14
5 (4 x 8)	102	2.04
2 (4 x 4)	121	2.42
7 (6 x 4)	532	10.6
6 (6 x 3)	664	13.3

As the matrix M becomes more "elongated," i.e., more jobs than machines, the probability of multiple optimal solutions increases and vice versa.

Effect of the Range of Operation Times

A comparison of the figures in Table I for Groups 1 and 8 and 4 and 6 does not indicate any appreciable effect when the range was changed from 1-9 to 1-19 on the same size problems.

Effect of Distribution

Two 5 x 5 groups (10 and 11) were tested in which operation times were drawn from a quasi-normal distribution in which the mean is ten and all entries were truncated to integer values. Note that as the distribution approached the uniform (group 11, variance = 4), the percentage of switches is approximately that of Groups 1 and 8. When the variance is diminished (Group 10, variance = 2) the percentage of switches decreases. We have

Group	n x m	Dist.	% Sw	NOP
1	5 x 5	Unif. 1,9	51.8	173
8	5 x 5	Unif. 1,19	49.4	154
11	5 x 5	N(10,4)	50.5	173
10	5 x 5	N(10,2)	39.3	264

Note that NOP also takes an appreciable jump. An explanation might be found in the extreme case of a variance equal to zero where all operations in M are of identical length. This results in all solutions becoming optimal and 0 per cent switches. Thus we can conclude that if operation times are normally distributed the number of optimal solutions (NOP) should be inversely proportional to the variance and the % switches directly proportional to the variance.

Some Overall Results

In almost every group, the percentage of operations that did <u>not</u> cause a switch was over 50%. These figures are on the conservative side in that a value of K = 10 was used in all cases.

For short operations, it is unlikely that an estimate would be in error by such a large percentage. For example, in Group 1, suppose we say an operation of time = 1-4 is counted as a non-switch if the time at which it does switch is greater than or equal to 10. Also count an operation of duration 5-9 as a non-switch if a switch occurs at a value of 15 or more. This would transfer 143 operations from the switch to the non-switch side of the ledger. The non-switch percentage goes from 48.2% to 59.6%. The point to be made is that a substantial proportion of operations do not cause a change in optimal sequence, within practical limits. It should be kept in mind, however, that we are only considering changes to one operation at a time.

Multiple Optimal Solutions

One might come to the conclusion that, if there are many optimal solutions, then max s_{ij} for all operations will be greater than zero. Actually this is not the case. An operation which is on the critical path ($s_{ij} = 0$) in more than one optimal schedule tends to stay on the critical path in the remainder of the optimal schedules. In only one instance out of 550 were all max $s_{ij} = 0$ for all i and j.

This is not to say that multiple optimal solutions provide no flexibility. For example, consider the matrices in Group 1. Eighteen out of 50 had a unique solution. These eighteen matrices averaged 11.3 operations on the overall critical path, i.e.,

max $s_{ij} = 0$. The 32 matrices with more than one optimal sequence averaged 7.7 operations on the overall critical path. It must be kept in mind that we are assuming that we can still change from one sequence to another, i.e., the schedule is not yet "in flight." There is considerable variation in the magnitude of positive slack among different optimal schedules which provides more flexibility.

All Critical Path Operations on a Machine

If an entire column of S_{max} is composed of all zero elements, there is a tendency for all operations in that column to fall into the non-switch column. For example, in Group 6 (6 x 3) there were 37 instances in which a column vector of all zeroes was observed in S_{max}. In 27 of these cases there were no switches on any of these operations. This accounts for 6 x 27 = 162 "zero" operations, a sizable amount. It should be recalled that "zeroes" in general had more of a tendency to switch. Lack of time has prevented further investigations of this apparent anomaly.

XIV. CONCLUSIONS

An investigation of this nature has as its goal the discovery of some rule or set of rules which are observed to be obeyed in a clear cut majority of instances. Unfortunately, evidence of such unequivocal rules was not observed. A factor of some significance turned out to be the length of an operation. The conclusions drawn relevant to this and other specific observations have been delineated in the previous section. In this section we shall make two further observations.

Usually, when one seeks a solution to a flowshop scheduling problem, the common practice is to stop after one optimal solution is discovered. With a little additional effort (at least when using a Branch and Bound algorithm) multiple optimal solutions can be found if they exist. This can provide a scheduler with considerable flexibility. It may enable him to select a schedule such that operations for which the estimates are most in doubt are placed so that they are given a maximum amount of slack. It may even be advantageous to select a sequence which is non-optimal (i.e., has a greater makespan) but has multiple optimal solutions (at that value of makespan) and hence more operations with more slack, i.e., more flexibility.

Muth[22] concludes that the schedule makespan is not very sensitive to moderately large errors in estimated operation times. Since over fifty per cent of the operations did not cause a switch in the

optimal sequence, we can conclude in a similar manner, that the original group of optimal sequences is not very sensitive to moderately large errors in estimated operation times when one of these times is increased while the others remain constant.

In other words, given the situation investigated by this thesis, the action, "do nothing," has a probability of better than one half of producing an optimal schedule for the revised problem. Any rescheduling should be carried out with this fact in mind.

XV. AREAS OF FURTHER INVESTIGATION

As direct extensions of this investigation are several questions: "When the optimal sequence does switch, what is the new optimal sequence?" Are the new optimal sequences formed by merely transposing two jobs? Or, are completely revised permutations produced? This type of investigation might provide some insight into the action to be taken by a scheduler when confronted with a revision to his original problem.

The entire section which discusses perturbations provides many additional areas of investigation and might be included under this heading. There has been very little published work on the rescheduling function of the static machine shop problem and this entire area appears to have many ramifications which can be analyzed. By selecting any static problem with its unique objective function and constraints, and hypothesizing a change in a priori information, we find an interesting possibility for extension of this work.

APPENDIX A

ABBREVIATIONS USED IN TABULATED RESULTS

Set # - Number assigned to set of ten input matrices produced by a separate random number "Seed".

NOP - Number of optimal solutions.

SW M1 - Number of "Switches" on Machine 1.

xxx SW - Number of switches of operations with values xxx.

SW - Switches.

NSW - Non Switches.

OPER. - Total operations.

Zero - Number of operations with a max. $S_{ij} = 0$, i.e., total number of critical path operations.

ZSW - Number of critical path operations that switch.

NZ - Number of operation with positive slack, i.e., total number of non-critical path operations.

NZSW - Number of non-critical path operations that switch.

GROUP 1

n x m = 5 x 5 Dist = Unif., r = 1, 9

Set #		1	2	3	4	5	Total
NOP		27	35	34	46	31	173
SW	M1	25	22	21	20	23	111
	M2	34	23	34	27	29	147
	M3	28	20	35	21	26	130
	M4	26	25	26	32	28	157
	M5	25	27	21	24	25	122
123 SW		43	48	63	53	58	265
456 SW		57	44	43	46	39	227
789 SW		42	25	31	27	31	156
SW		141	117	137	124	124	648
NSW		109	133	113	126	126	602
OPER		250	250	250	250	250	1250
ZERO		83	93	74	98	102	450
Z SW		56	55	47	64	62	284
NZ		167	157	176	152	148	800
NZSW		85	62	90	60	62	364

GROUP 2

n x m = 4 x 4 Dist = Unif., r = 1, 9

Set #		1	2	3	4	5	Total
NOP		23	17	25	26	30	121
SW	M1	13	17	15	10	17	72
	M2	16	15	16	13	22	72
	M3	10	13	15	19	19	76
	M4	15	17	19	15	18	84
123	SW	27	32	38	31	39	147
456	SW	20	17	13	19	26	95
789	SW	7	12	14	7	10	50
SW		54	62	65	57	66	304
NSW		106	98	95	103	94	496
OPER		160	160	160	160	160	800
ZERO		65	75	72	65	62	339
2	SW	28	36	35	25	31	155
NZ		95	85	88	95	98	461
NZSW		26	26	30	32	35	149

GROUP 3

n x m = 3 x 3 Dist = Unif.ρ r = 1, 9

Set #		1	2	3	4	5	Total
NOP		13	12	16	12	13	66
SW	M1	10	6	6	6	9	37
	M2	6	4	8	6	1	25
	M3	8	13	5	8	5	39
123	SW	13	16	14	11	12	66
456	SW	10	5	5	9	1	30
789	SW	1	2	0	0	2	5
SW		24	23	19	20	15	101
N SW		66	67	71	70	75	349
OPER		90	90	90	90	90	450
ZERO		52	51	52	60	55	271
ZERO	SW	20	17	14	17	12	80
N ZERO		38	39	38	30	35	179
N Z	SW	4	6	5	3	3	21

GROUP 4

n x m = 4 x 6 Dist = Unif., r = 1, 9

Set #		1	2	3	4	5	Total
NOP		18	24	18	25	22	107
SW	M1	21	11	14	17	16	79
	M2	17	14	19	12	23	85
	M3	20	14	17	12	17	82
	M4	19	11	21	12	19	82
	M5	18	12	15	17	21	83
	M6	11	19	16	18	19	83
123 SW		39	38	53	33	52	212
456 SW		45	21	26	38	41	171
789 SW		23	22	23	17	21	106
SW		106	81	103	88	115	493
NSW		136	159	137	152	125	707
OPER		240	240	240	240	240	1200
ZERO		83	87	72	91	90	423
2 SW		38	41	30	47	50	206
N Z		167	153	168	149	150	777
N Z SW		68	40	73	41	65	187

GROUP 5

n x m = 4 x 8 Dist = Unif., r = 1, 9

Set #		1	2	3	4	5	Total
NOP		23	22	22	20	15	102
SW	M1	21	19	17	22	19	98
	M2	21	19	17	22	19	98
	M3	13	15	17	21	17	83
	M4	11	14	16	18	16	75
	M5	13	16	13	11	16	69
	M6	13	15	17	21	17	83
	M7	16	18	17	18	20	89
	M8	20	20	16	16	18	90
123 SW		53	55	47	63	56	273
456 SW		43	50	40	57	50	240
789 SW		33	31	38	30	36	168
SW		128	136	130	149	142	685
N SW		192	184	190	171	178	915
OPER		320	320	320	320	320	1600
ZERO		86	123	102	131	119	551
2 SW		46	62	55	65	57	285
N Z		234	197	218	189	201	1049
N Z SW		82	74	75	84	85	400

GROUP 6

n x m = 6 x 3 Dist = Unif., r = 1, 9

Set #		1	2	3	4	5	Total
NOP		135	155	117	113	144	664
SW	M1	20	25	26	33	31	125
	M2	18	15	30	21	24	108
	M3	26	30	22	25	27	130
123 SW		28	41	35	37	39	180
456 SW		30	14	29	26	23	122
789 SW		6	13	14	16	20	69
SW		64	70	78	79	82	373
NSW		116	110	102	101	98	529
OPER		180	180	180	180	180	900
ZERO		88	86	91	87	91	443
2 SW		32	26	42	41	35	176
N Z		92	94	89	93	89	457
N Z SW		32	44	36	38	47	197

GROUP 7

n x m = 6 x 4 Dist = Unif., r = 1, 9

Set #		1	2	3	4	5	Total
NOP		79	101	73	172	107	532
SW	M1	21	23	27	29	22	122
	M2	24	24	30	31	42	151
	M3	37	31	28	31	31	158
	M4	29	36	27	18	22	132
123	SW	56	57	48	52	53	266
456	SW	33	38	38	32	46	187
789	SW	22	19	26	29	18	114
SW		111	114	112	109	117	567
NSW		139	136	138	141	133	633
OPER		240	240	240	240	240	1200
ZERO		102	113	92	115	86	508
2 SW		50	55	55	76	49	285
NZ		138	137	148	135	154	692
N Z SW		61	59	57	33	68	282

GROUP 8

n x m = 5 x 5 Dist = Unif., r = 1, 19

Set #		1	2	3	4	5	Total
NOP		34	31	26	29	34	154
SW	M1	20	22	27	20	27	116
	M2	17	28	31	29	21	126
	M3	24	27	29	33	25	138
	M4	21	25	24	29	26	125
	M5	19	21	26	23	23	112
1-6 SW		37	52	48	54	48	239
7-12 SW		33	42	51	40	43	208
13-19 SW		31	29	38	40	31	168
SW		101	123	137	134	122	617
NSW		149	127	113	116	128	633
OPER		250	250	250	250	250	1250
ZERO		94	79	87	96	84	460
2 SW		45	46	58	49	46	244
N Z		156	171	163	154	166	790
N Z SW		56	77	79	85	76	373

GROUP 9

n x m = 4 x 6 Dist = Unif., r = 1, 19

Set #		1	2	3	4	5	Total
NOP		24	21	16	14	21	96
SW	M1	14	15	13	22	11	75
	M2	10	23	10	16	15	74
	M3	12	18	13	17	14	74
	M4	14	16	10	19	20	79
	M5	20	12	15	20	21	88
	M6	24	17	15	24	19	89
1-6 SW		41	40	37	39	42	199
7-12 SW		26	23	23	38	33	143
13-19SW		27	38	16	41	25	147
SW		94	101	76	118	100	489
NSW		146	139	164	132	140	711
OPER		240	240	240	240	240	1200
ZERO		91	90	94	111	87	473
2 SW		46	42	34	72	36	230
N Z		149	150	146	129	153	727
N Z SW		48	48	42	46	64	259

GROUP 10

n x m = 5 x 5 Dist = N (10, 2)

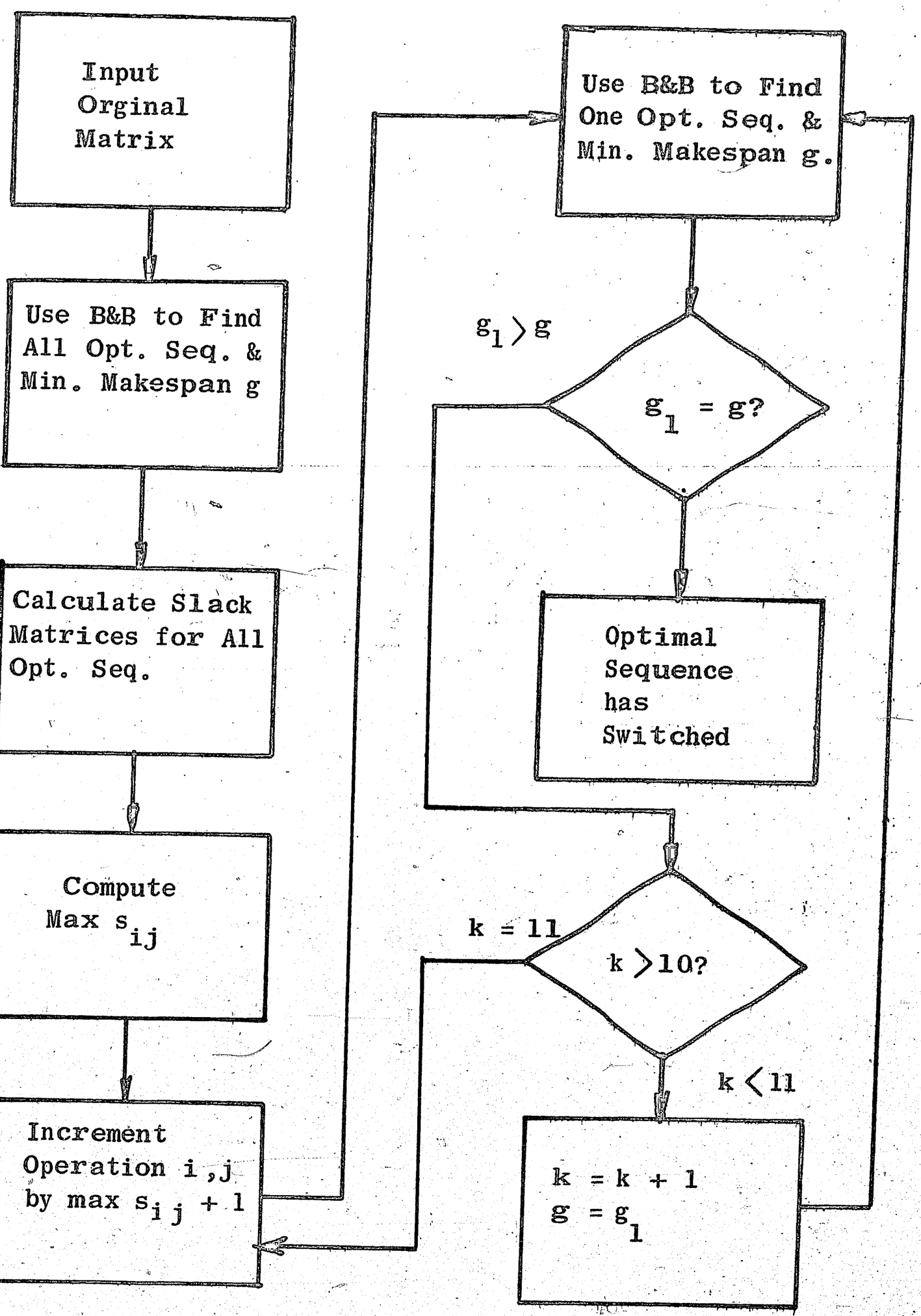
Set #		1	2	3	4	5	Total
NOP		106	45	51	46	22	264
SW	M1	20	12	16	26	21	95
	M2	17	11	13	21	23	85
	M3	23	12	12	23	24	94
	M4	21	26	14	22	31	114
	M5	27	19	13	20	24	103
8 SW		39	32	23	43	41	178
9,10,11 SW		50	36	41	52	66	245
12 SW		19	12	4	17	16	68
SW		108	80	68	112	123	491
N SW		142	170	182	138	127	759
OPER		250	250	250	250	250	1250
ZERO		93	77	78	90	97	445
2 SW		44	33	30	51	58	216
N Z		157	173	172	160	153	805
N Z SW		64	47	38	61	67	275

GROUP 11

n x m = 5 x 5 Dist = N (10, 4)

Set #		1	2	3	4	5	Total
NOP		17	15	51	25	65	173
SW	M1	25	26	20	23	24	118
	M2	25	28	25	29	22	129
	M3	24	16	23	31	16	130
	M4	23	29	30	34	15	132
	M5	19	28	27	34	14	122
8 SW		50	52	53	73	45	273
9,10,11 SW		45	40	38	46	24	193
12 SW		21	35	34	32	22	145
SW		116	127	125	151	91	631
N SW		134	123	125	99	159	619
OPER		250	250	250	250	250	1250
ZERO		109	95	94	86	90	384
2 SW		60	58	51	54	47	270
N Z		141	155	156	164	160	866
N Z SW		56	69	74	97	44	361

APPENDIX B



SIMPLIFIED BLOCK DIAGRAM OF MAIN PROGRAM

```
      INTEGER G(10,10),P(10),F(10,10,10),TOTMT(100),TOTJR(10,9),GP(10,10
     C),R,OLDN,TOTJF,SLACK(10,10),OPSEQ(10,20,11)                          100
      DIMENSION MATRX(10,10),NODE(10),MARTY(10,10),MAXX(10,10),JOBVL(10,   101
     C10)
 4001 READ(1,1)NUMM,NUMJ,IX,LOTIM,IUPTM,MXNDE,NUMMAT
    1 FORMAT(2I5,I9,4I5)
      IF(NUMM)999,491,999
  999 N=0
 4000 N=N+1                                                               105
      DO 1000 J=1,NUMM
      DO 1000 I=1,NUMJ
      CALL RANDU(IX,IY,YFL)
      IX=IY
      MATRX(I,J)=YFL*(IUPTM-LOTIM)+LOTIM
 1000 CONTINUE
      KT=0
      MING1=10**4                                                          110
      NOP=0                                                                111
      KT2=1                                                                112
      NA=1                                                                 113
      NB=1                                                                 114
      MMS=0                                                                115
  560 KT=KT+1                                                              116
      GO TO(1001,630),KT2                                                  117
 1001 MMS=MMS+1                                                            118
      NDECN=0                                                              119
      DO 3 J=1,NUMM                                                        120
      TOTMT(J)=0                                                           121
      DO 3 I=1, NUMJ                                                       122
      F(I,1,J)=MATRX(I,J)                                                  123
      TOTMT(J)=TOTMT(J)+F(I,1,J)                                           124
    3 CONTINUE                                                             125
      DO 30 I=1,NUMJ                                                       126
      TOTJR(I,NUMM-1)=F(I,1,NUMM)                                          127
      INUM=NUMM-2                                                          128
                                                                           129
```

```
      DO 30 J=1,INUM
      MM=NUMM-J
      TOTJR(I,MM-1)=TOTJR(I,MM)+F(I,1,MM)
30    CONTINUE
      DO 40 I=1,NUMJ
      DO 40 J=1,NUMM
      IF(J-NUMM)39,41,39
39    MINJR=10**4
      DO 42 K=1,NUMJ
      IF(K-I)38,42,38
38    IF(MINJR-TOTJR(K,J))42,42,37
37    MINJR=TOTJR(K,J)
42    CONTINUE
      GO TO 43
41    MINJR=0
43    TOTJF=0
      IF(J-1)36,44,36
36    IINUM=J-1
      DO 45 L=1,IINUM
      TOTJF=TOTJF+F(I,1,L)
45    CONTINUE
44    GP(I,J)=TOTJF+TOTMT(J)+MINJR
40    CONTINUE
      DO 46 I=1,NUMJ
      G(I,1)=GP(I,1)
      DO 46 J=2,NUMM
      IF(G(I,1)-GP(I,J))35,46,46
35    G(I,1)=GP(I,J)
46    CONTINUE
      MING=G(1,1)
      NODE(1)=1
      DO 47 I=2,NUMJ
      IF(G(I,1)-MING)48,47,47
48    MING=G(I,1)
      NODE(1)=I
```

```
47      CONTINUE
        NDECN=NDECN+1
  191   MINES=MING1
        II=1
   51   MS=NODE(II)
        F(MS,II,1)=-1
        P(1)=MATRX(MS,1)
        DO 50 J=2,NUMM
        P(J)=P(J-1)+MATRX(MS,J)
   50   CONTINUE
   85   II=II+1
        R=0
   90   R=R+1
   95   IF(R-NODE(II-1))96,110,96
   96   IF(F(R,II-1,1))115,53,53
   53   F(R,II,1)=P(1)+MATRX(R,1)
        DO 55 J=2,NUMM
        IF(F(R,II,J-1)-P(J))54,52,52
   52   F(R,II,J)=F(R,II,J-1)+MATRX(R,J)
        GO TO 55
   54   F(R,II,J)=P(J)+MATRX(R,J)
   55   CONTINUE
        IF(II-NUMJ)183,184,183
  183   DO 101 J=1,NUMM
        TOTND=0
        MINJR=10**4
        DO 103 I=1,NUMJ
        IF(R-I)230,103,230
  230   IF(I-NODE(II-1))231,103,231
  231   IF(F(I,II-1,1))103,232,232
  232   TOTND=TOTND+MATRX(I,J)
        IF(J-NUMM)233,103,233
  233   IF(MINJR-TOTJR(I,J))103,103,234
  234   MINJR=TOTJR(I,J)
  103   CONTINUE
```

```
      IF(J-NUMM)105,104,105
104   MINJR=0
105   GP(R,J)=F(R,II,J)+TOTND+MINJR
101   CONTINUE
      G(R,II)=GP(R,1)
      DO 120 J=2,NUMM
      IF(G(R,II)-GP(R,J))235,120,120
235   G(R,II)=GP(R,J)
120   CONTINUE
      IF(R-NUMJ)90,107,107
107   MING=10**4
      NODE(II)=1
      DO 150 J=1,NUMJ
      IF(MING-G(J,II))150,150,236
236   MING=G(J,II)
      NODE(II)=J
150   CONTINUE
      NDECN=NDECN+1
      GO TO(237,500),KT
237   IF(MINES-MING)160,160,152
500   IF(MINES-MING)160,152,152
152   DO 153 J=1,NUMM
      MN=NODE(II)
      P(J)=F(MN,II,J)
153   CONTINUE
      F(MN,II,1)=-1
      GO TO 85
160   II=II-1
      MO=NODE(II)
      G(MO,II)=10**4
      GO TO 195
184   G(R,II)=F(R,II,NUMM)
      MING=G(R,II)
      NODE(II)=R
      IF(MINES-MING)505,505,238
```

200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234

```
238 MINES=MING                                                          235
505 MING1=MING                                                          236
    GO TO(185,540),KT                                                   237
540 NOP=NOP+1                                                           238
    IF(NOP-20)541,541,556                                               239
556 NOP1=20                                                             240
    GO TO 185                                                           241
541 NOP1=NOP                                                            242
    NH=1                                                                243
    CALL GANTT(NUMJ,NUMM,NODE,MATRX,SLACK,NH)                           244
    DO 550 I=1,NUMJ                                                     245
550 OPSEQ(I,NOP,1)=NODE(I)                                              246
    NUMM1=NUMM+1                                                        247
    DO 555 I=1,NUMJ                                                     248
    DO 555 K=2,NUMM1                                                    249
    K1=K-1                                                              250
    OPSEQ(I,NOP,K)=SLACK(I,K1)                                          251
555 CONTINUE                                                            252
185 II=II-1                                                             253
195 R=0                                                                 254
190 R=R+1                                                               255
    GO TO(510,520),KT                                                   256
510 IF(MINES-G(R,II))200,200,239                                        257
520 IF(MINES-G(R,II))200,239,239                                        258
239 IF(F(R,II,1))200,209,209                                            259
200 IF(R-NUMJ)190,240,190                                               260
240 IF(II-1)185,530,185                                                 261
530 GO TO(560,565),KT                                                   262
565 GO TO(600,630),KT2                                                  263
600 IF(MMS-2)601,601,602                                                264
601 CONTINUE                                                            265
    WRITE(3,16)
16  FORMAT(1H1,' THE JOB-TIME MATRIX FOR THIS RUN IS AS FOLLOWS ')      267
    WRITE(3,17)
17  FORMAT(1HO,' JOB NO.   M1   M2   M3   M4   M5   M6   M7   M8')    269
```

```
      DO 2000 I=1,NUMJ                                                  270
      WRITE(3,2)I,(MATRX(I,J),J=1,NUMM)
2     FORMAT(1H ,2X,I3,3X,20I5)
2000  CONTINUE
      WRITE(3,18) (TOTMT(J),J=1,NUMM)                                   272
18    FORMAT(1HO,'TOTALS   ',20I5)                                      273
      WRITE(3,5)
    5 FORMAT(1HO,' THE OPTIMAL SEQUENCES ARE')                          275
      DO 580 J=1,NOP1
      WRITE(3,570)(OPSEQ(I,J,1),I=1,NUMJ)                               277
  570 FORMAT(10X,10I3)                                                  278
  580 WRITE(3,590)
  590 FORMAT(1HO)                                                       280
      WRITE(3,595) MING1
  595 FORMAT(1HO,' THE OPTIMAL WORK DURATION IS',I3,/)                  282
      WRITE(3,596)
  596 FORMAT(1HO,' THE SLACK MATRICES FOR THE OPTIMAL SEQUENCES ARE')   284
      DO 503 J=1,NOP1                                                   286
      DO 502 I=1,NUMJ                                                   287
      WRITE(3,506)(OPSEQ(I,J,K),K=1,NUMM1)                              288
  506 FORMAT(10X,I2,5X,10I3)
  502 CONTINUE
  503 WRITE(3,504)                                                      291
  504 FORMAT(1HO,//)                                                    292
      WRITE(3,833)NOP
  833 FORMAT(' NBR OF OPTIMAL SOLUTIONS =',I5)                          294
      MING2=MING1
  602 NOLD=MATRX(NA,NB)
      MING3=0                                                           295
      NT=0                                                              296
      NB1=NB+1                                                          297
      MAXSL=0                                                           298
      DO 610 I=1,NOP1                                                   299
      IF(MAXSL-OPSEQ(NA,I,NB1))609,610,610                              300
  609 MAXSL=OPSEQ(NA,I,NB1)                                             301
                                                                        302
                                                                        303
```

```
610 CONTINUE
    MAXX(NA,NB)=MAXSL
    NE=MATRX(NA,NB)+MAXSL
615 LMING=MING1
    NE=NE+1
    MATRX(NA,NB)=NE
    NT=NT+1
    IF(NT-11)620,611,611
611 CONTINUE
    MARTY(NA,NB)=10000
    JOBVL(NA,NB)=10000
651 CONTINUE
    MING1=MING2
    MATRX(NA,NB)=NOLD
    IF(NB-NUMM)481,482,482
481 NB=NB+1
    KT2=1
    GO TO 602
482 NA=NA+1
    NB=1
    IF(NA-NUMJ)483,483,585
483 KT2=1
    GO TO 602
620 MING1=MING1+1
    KT2=2
640 NOP=0
    KT=1
    GO TO 1001
630 IF(MING1-LMING)635,635,632
632 CONTINUE
    GO TO 615
635 CONTINUE
    IF(MING1-MING3)615,615,670
670 CONTINUE
```

```
      MING3=MING1
      MARTY(NA,NB)=MATRX(NA,NB)
      JOBVL(NA,NB)=NOLD
      GO TO 651
209   OLDN=NODE(II)
210   MING=G(R,II)
      NODE(II)=R
  220 IF(R-NUMJ)242,215,242
  242 R=R+1
      IF(MING-G(R,II))220,220,243
  243 IF(F(R,II,1))220,210,210
215   F(OLDN,II,1)=1
      G(OLDN,II)=10**4
      NDECN=NDECN+1
  244 IF(II-1)152,51,152
110   F(R,II,1)=-1
      G(R,II)=10**4
      IF(R-NUMJ)245,107,245
  245 R=R+1
      GO TO 96
115   F(R,II,1)=-1
      G(R,II)=10**4
      IF(R-NUMJ)246,107,246
  246 R=R+1
      GO TO 95
  585 CONTINUE
      WRITE(3,590)
      DO 9000 I=1,NUMJ
      WRITE(3,934)I,(MARTY(I,J),J=1,NUMM)
  934 FORMAT(10X,I3,3X,10(2X,I2))
 9000 CONTINUE
      WRITE(3,590)
      DO 9002 I=1,NUMJ
      WRITE(3,934)I,(JOBVL(I,J),J=1,NUMM)
 9002 CONTINUE
      WRITE(3,590)
```

348

350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371

```
      DO 9001 I=1,NUMJ
      WRITE(3,934)I,(MAXX(I,J),J=1,NUMM)
 9001 CONTINUE
      IF(N.NE.NUMMAT)GO TO 4000
      GO TO 4001
  491 CONTINUE
      CALL EXIT
      END
      SUBROUTINE RANDU(IX,IY,YFL)
      IY=IX*65539
      IF(IY)300,305,305
  300 IY=IY+2147483647+1
  305 YFL=IY
      YFL=YFL*.4656613E-9
      RETURN
      END
      SUBROUTINE GANTT(NUMJ,NUMM,NODE,JTIME,SLACK,NH)
      INTEGER GANT(10,200),SLACK(10,10),CM,TJ,PS,CJ,CT,CJT,TTLJ,WT
      DIMENSION NODE(10),JTIME(10,10)
      LT2=1
      DO 5 I=1,10
      DO 5 J=1,200
    5 GANT(I,J)=0
      ITT=0
      DO 130 I=1,NUMM
      LTI=LT2
      DO 130 KK=1,NUMJ
      K=NODE(KK)
      LT=LTI
      LEND=JTIME(K,I)+LTI-1
      DO 110 J=LT,LEND
      LTI=LTI+1
      IF(I-1)100,100,20
   20 II=I
   21 II=II-1
```

373
374
375

```
      IF(II)22,100,22
   22 IF(GANT(II,J))23,21,23
   23 IF(GANT(II,J)-K)100,30,100
   30 LEND=LEND+1
      GO TO 110
  100 GANT(I,J)=K
  110 CONTINUE
      IF(KK-1)130,120,130
  120 LT2=LTI
  130 CONTINUE
      LTI=LTI-1
      CM=NUMM
      TJ=1
      PS=0
      CJ=NODE(NUMJ)
      SLACK(CJ,CM)=0
      CT=LTI
  303 CJT=JTIME(CJ,CM)
      CT=CT-CJT+1
      IT=0
  300 IT=IT+1
      ITT=ITT+1
      CT=CT-1
      IF(GANT(CM,CT))301,300,301
  301 CJ=GANT(CM,CT)
      ITT=ITT-1
      TJ=TJ+1
      SLACK(CJ,CM)=IT-1+PS
      PS=PS+IT-1
      IF(TJ-NUMJ)303,304,304
  304 CM=CM-1
      CJ=NODE(NUMJ)
      TTLJ=0
      IT=0
      I=NUMM+1
```

```
305 I=I-1
    TTLJ=TTLJ+JTIME(CJ,I)
    IF(I-CM-1)318,318,305
318 CT=LTI-TTLJ+1
306 CT=CT-1
    IT=IT+1
    IF(GANT(CM,CT))306,306,307
307 SLACK(CJ,CM)=IT-1
    TJ=1
316 IT=0
    CT=CT-JTIME(CJ,CM)+1
309 IT=IT+1
    ITT=ITT+1
    CT=CT-1
    IF(GANT(CM,CT))309,309,310
310 IT=IT-1
    ITT=ITT-1
    NJ=CJ
    CJ=GANT(CM,CT)
    MIN1=IT+SLACK(NJ,CM)
    WT=0
    I=CT
312 I=I+1
    WT=WT+1
    IF(CJ-GANT(CM+1,I))312,311,312
311 MIN2=WT-1+SLACK(CJ,CM+1)
    IF(MIN1-MIN2)314,314,313
313 SLACK(CJ,CM)=MIN2
    GO TO 315
314 SLACK(CJ,CM)=MIN1
315 CONTINUE
    TJ=TJ+1
    IF(TJ-NUMJ)316,317,317
317 IF(CM-1)320,320,304
320 CONTINUE
```

80

```
      WRITE(3,367)ITT
  367 FORMAT(1HQ,' TOT IDLE TIME =',I3)
      NJS=0
      NTS=0
      DO 360 K=1,NUMJ
      DO 360 L=1,NUMM
      IF(SLACK(K,L))360,360,355
  355 NJS=NJS+1
      NTS=NTS+SLACK(K,L)
  360 CONTINUE
      WRITE(3,365)NJS,NTS
  365 FORMAT(' TOT OPER WITH SLACK =',I2,' TOT SLACK =',I4)
      XNTS=NTS
      XNJS=NJS
      AVG=XNTS/XNJS
      WRITE(3,366)AVG
  366 FORMAT(' AVG SLACK PER OPER =',F5.2)
      GO TO(350,351),NH
  350 CONTINUE
  351 CONTINUE
      RETURN
      END
```

BIBLIOGRAPHY

1. Akers, S. B., "A Graphical Approach to Production Scheduling Problems," Operations Research, Vol. 4, No. 2, April 1956.

2. Agin, N. "Optimum Seeking with Branch and Bound," Management Science, Vol. 13, No. 4, December 1966.

3. Conway, R. W., "Priority Dispatching and Work-in-Process Inventory in a Job Shop," J. Ind. Eng., Vol. 16, No. 2, March 1965.

4. Conway, R. W., "Priority Dispatching and Job Lateness in a Job Shop," J. Ind. Eng., Vol. 16, No. 4, July 1965.

5. Conway, R. W. and Maxwell, W. L., "Network Dispatching by the Shortest Operation Discipline," Operations Research, Vol. 10, No. 1, February 1962.

6. Conway, R. W., Maxwell, W. L. and Miller, L. W., Theory of Scheduling, Addison Wesley Publishing Co., 1967.

7. Dudek, R. A. and Teuton, O. F., "Development of M-Stage Decision Rule for Scheduling n Jobs Through m Machines," Operations Research, Vol. 12, No. 3, May 1964.

8. Gere, W., "A Neuristic Approach to Job Shop Scheduling," Management Science, Vol. 13, No. 3, November 1966.

9. Giglio, R. J. and Wagner, H. M., "Approximate Solutions to the Three-Machine Scheduling Problem," Operations Research, Vol. 12, No. 2, March 1964.

10. Heller, J., "Some Numerical Experiments for an M J Flow Shop and Its Decision Theoretical Aspects," Operations Research, Vol. 8, No. 2, March 1960

11. Ignall, E. and Schrage, L., "Application of the Branch and Bound Technique to Some Flow Shop Scheduling Problems," Operations Research, Vol. 13, No. 3, May 1965.

12. Jackson, J. R., "An Extension of Johnson's Results on Job Lot Scheduling," Nav. Res. Log. Quart., Vol. 3, No. 3, September 1956.

13. Johnson, S. M., "Optimal Two and Three Stage Producion Schedules with Setup Times Included," Nav. Res. Log. Quart., Vol. 1, No. 1, March 1954.

14. Karush, W., "A Counterexample to a Proposed Algorithm for Optimal Sequencing of Jobs," Operations Research, Vol. 13, No. 2, March 1965.

15. Lawler, E. L. and Wood, D. E., "Branch and Bound Methods: A Survey," Operations Research Quarterly, Vol. 14, 1966.

16. Little, J. D. C., Murty, K. G., Sweeney, D. W. and Karel, C., "An Algorithm for the Traveling Salesman Problem," Operations Research, Vol. 11, No. 6, November 1963.

17. Lomnicki, Z. A., "A Branch and Bound Algorithm for the Exact Solution of the Three Machine Scheduling Problem," Operations Research Quarterly, Vol. 16, 1965.

18. Lomnicki, Z. A. and Brown, A. P. G., "Some Applications of the Branch and Bound Algorithm to the Machine Scheduling Problem," Operations Research Quarterly, Vol. 17, 1966.

19. Long, T. B., Research Engineer, Western Electric Company, Inc., Private Communication.

20. Manne, A. S., "On the Job Shop Scheduling Problem," Operations Research, Vol. 8, No. 2, March 1960.

21. McMahon, G. B. and Burton, P. G., "Flow Shop Scheduling with the Branch and Bound Method", Operations Research, Vol. 15, No. 3,

22. Muth, J. F. and Thompson, G. L., Industrial Scheduling, Prentice-Hall, 1963.

23. Wagner, H. M., "An Integer Linear-Programming Model for Machine Scheduling," Nav. Res. Log. Quart., Vol. 6, No. 2, June 1959.

VITA

PERSONAL HISTORY

Name: Milan Martin Sowis

Date of Birth: September 19, 1936

Place of Birth: Newark, New Jersey

Parents: Martin and Mary Sowis

EDUCATIONAL BACKGROUND

Irvington High School Irvington, New Jersey	Graduated - 1954
Stevens Institute of Technology Hoboken, New Jersey	Graduated - 1958
Lehigh University Candidate for Master of Science in Industrial Engineering	1967-1969

HONORS

Alpha Pi Mu Fraternity

PROFESSIONAL EXPERIENCE

Systems Equipment Engineer Western Electric Co., Inc. Newark, New Jersey	November 1958 - July 1965
Planning Engineer Western Electric Co., Inc. New York City	July 1965 - July 1967
Information Systems Staff Member Western Electric Co., Inc. Princeton, New Jersey	July 1967 - June 1969